CODI - A System for Cooperative 3D Design

R. Galli*, P. Palmer*, M. Mascaro*, M. Dias**, Y. Luo*
* Departamento de Matemáticas e Informática
Universitat de Les Illes Balears
070701 Palma de Mallorca, Spain
e-mail: {gallir, pere, mascport, yuhua}@anim.uib.es

** ADETTI/ISCTE, Av. das Forças Armadas 1600 Lisboa, Portugal
e-mail:jmd@adetti.iscte.pt

Abstract

A system that integrates 3D virtual scene generation tools and data communication techniques for 3D cooperative design work is presented in the paper. The named system CODI (*sistema COoperativo de DIseño3D*) supports two major functions: 3D cooperative design and multiple client distant access. The system is implemented using Open Inventor Toolkit in C++ with VRML compatibility.

Keywords 3D Interactive Graphics, CSCW, Cooperative 3D design

1. Introduction

This paper presents a CSCW system for cooperative 3D object design. Each of the designers can perform the design work in his own location together with other designers for a common virtual 3D world at the same time. The graphics workstations used in the design work will be connected together by communication network. The system has two major functions. The first is to support the real time cooperative working session for 3D design and the second is to support the multiple client distant access to the system.

The application of the system can be very wide such as in virtual studio, cooperative-CAD/CAM etc. The system will allow several system users to add, modify, or delete objects, change environment attributes, traverse the common environment remotely and interactively. The change by any user would be visible to other distant users in the group. They may obtain a 3D virtual scene, an architecture design or their elements from remote locations just as if they were local. The clients of the system can access the system from multiple locations remotely and interactively.

The system presented here is named *CODI (sistema COoperativo de DIseño3D*). When stands alone, the system has all the major functions for single user 3D design work such as authoring, editing and visualization. When a cooperative working session is called, the system turns to a real time co-working mode for a group of users to work at different locations together. Video and audio conferencing facilities will also be available for the cooperative working session.

There are two major blocks in the system. One is the application block and the other is the network communication and cooperative support block. The application block hosts the current major application -the 3D design tool. The network communication and cooperative support block provides the session control and the communication between cooperative working machines.

The paper is organized as follows. Section 2 discusses the CODI system design consideration. Section 3 describes the network communication and cooperative support platform. Section 4 introduces the 3D design block supported by the service provided by the communication block. Section 5 concludes the paper.

2. System Consideration

In general, the design of multi-user collaborative environment is based on two main approaches. One is based on dedicated groupware system which only supports a particular application or a set of particular applications. The other provides generic support services for a wide range of distributed applications. The latter provides more flexibility to create shared environments with distinguished applications using the same support platform. The advantage of separating the application semantics from the communication platform is that any additional control and service functions to cope with future network technologies will not affect the application themselves. The design of

IEEE Conference on Information Visualization (IV '97)
1093-9547/97 $10.00 © 1997 IEEE

our network communication and cooperative support block is based on the second approach.

2.1 The Communication Support

Two different types of services are provided by the network communication and cooperative support block: Session Control and Communication. These services are available to the applications (here the 3D design system) through a common service interface.

Cooperative support platforms may rely on centralized or distributed configurations [1]. Hybrid schemes may also be used. A fully distributed configuration[2] is accepted in our system with application resources available on all the sites. In this context, an application entity exists in each site. The platform supports the distributed interaction among all application instances. The messages generated by one application replica are transmitted to other sites sharing the same application environment. In most of the transactions the messages have the same semantic contents. Cooperative applications usually have a consistency requirement, i.e. all the sites must have the same data state. The multi-user 3D design system in our case has a strong requirement on the consistency. The messages exchanged among the sites must arrive at each end-point with the same order as they are produced. This implies a synchronous data exchange requirement. Event-driven synchronization sheme is used in our system for consistency keeping instead of using globalsystem time.

In addition to communication service multiplexing, the platform should also have session control mechanisms for the supported client applications. The relevant issues include the QoS provision (Quality of Service), consistency control, new member admission into a running session, new distributed application invocation, exceptional event and failures handling, definition of roles within the group etc.

Different types of access control over shared objects should be established according to the set of consistency mechanisms,. For example, during a joint 3D navigation session, the users may change a conservative locking policy into a more open one or may desire to re-define the policy[4].

The member admission into a running session requires the availability of session control and communication resources on the new user's site. At the application level, the integration of a new member requires additional application resources and a state update procedure for each shared application. Specific tasks include the exchange of files and other shared data structures and also changes in the visualization process on each site.

2.2 The System Overview

The 3D design application is currently the major application in the application block. As a 3D design tool, it should provide all the editing, visualization capability to the 3D scene and its elements, the environment attributes for both single user mode and the cooperative sharing mode. The communication mechanism underneath should be hidden from the users.

As a result, a full distributed architecture has been designed for the system. The functional modules of the CODI system are shown in figure 1 which is a group of member workstations connected by the network.

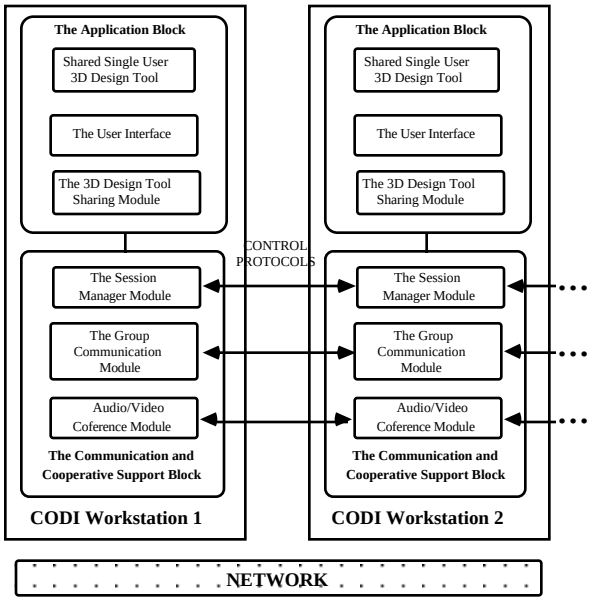

Figure 1: Global structure of CODI system.

The Network Communication and Cooperative Support block provides the cooperative working session control and the communication between cooperative working machines.

3. The Network Communication and Cooperative Support Block

From the point of view of communication and cooperative support, the CODI system is a layered structure as shown in figure 2. This is the structure in one member workstation. The whole configuration is a group of such workstations connected by the network.

3.1 The layered architecture.

As shown in figure 2, there are basically three layers for communication and control: The Application Environment Layer, Session Manager Layer and the Group

Communication Layer. The highest layer, the application layer holds the cooperative application programs for the cooperative 3D design and belong in our explanation to the Application Block.

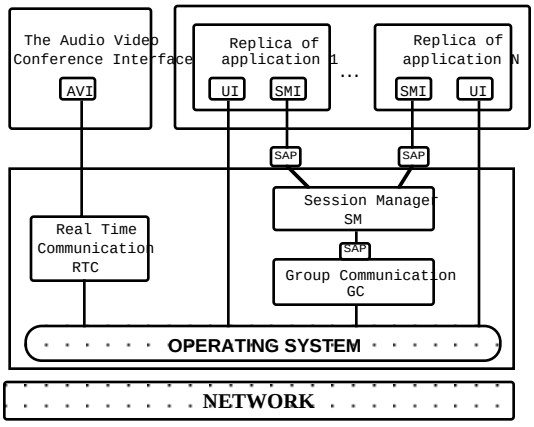

Figure 2. The layered structure from the communication point of view

The Session Manager (SM) layer is responsible for the cooperative working session control tasks. The Group Communication (GC) layer is responsible for point-to-multipoint or point-to-point communication functions, with a specific quality of service (QoS). These two entities are the building modules of the communication and cooperative support block. In a more general case, the application environment layer encompasses several distributed applications that act as clients requesting service to the platform through the Service Access Points (SAPs) created on top of the SM service interface. What the communication and cooperative support block can see in the application block are only two kinds of elements: the user interface (UI) and the session manager interface (SMI) of the replica of an application. Notice that, such platform can support more than one applications on each site so long as each has its own UI and SMI. The structure in figure 3 on each site is replicated. Session control is achieved by a distributed protocol executed among all the SM entities with the same capabilities. Some specific attributes may be given to just one of the session managers, to perform special control tasks such as the inclusion of users or applications.

A distributed application performs a specific protocol in order to exchange messages. Most of the messages contain user events produced through one or more input devices. These local events are encapsulated in messages and sent to the other sites through a service request made to the SM entity. The model in figure 2 contains two I/O logical entities located at the application layer environment. One of them is the User Interface (UI), which is responsible for collecting local events in the context of a particular application (audio and video devices generate events with real-time requirements). The Session Manager Interface (SMI) structures the communication between the application layer and the SM layer. It is the interface between applications and the support platform.

3.2 Layered Control Protocols

The major functions to support the cooperative design are realized by implementing a set of protocols at each layer, as shown in figure 3. The same layers at different participant's machines will communicate with each other by the same protocol.

At the highest layer, the application layer, the Session Manager Interfaces (SMI) in the application block in each replica of the participant machines perform a specific application layer protocol, the SMI protocol. They use the service supplied by their Session Manager in the communication block. Messages created at this layer are tele-events which require a reliable network service, and real-time events such as tele-virtual-cameras which require a best effort connection.

The Session Managers SM at all participant sites exchange messages through a SM layer protocol. The messages at this layer contain either application layer messages (to be delivered at the destination to the SMI entities), or session control messages whose destination is the set of other peer SM entities at the same layer. The SM layer uses the services provided by the GC layer interface.

Group Communication (GC) is achieved by a distributed GC protocol executed by all the GC entities at each distributed site. This layer hides the multipoint configuration from the SM layer including the set of network or transport protocols used and specific network technology. The messages exchanged among the GC entities contain either SM data or specific protocol messages whose destination is the set of peer GC entities itself. The GC layer uses the available network and/or transport protocols.

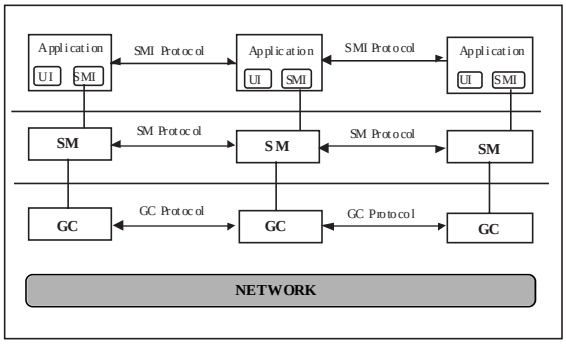

Figure 3 Protocols for layered distributed interactions.

IEEE Conference on Information Visualization (IV '97)
1093-9547/97 $10.00 © 1997 IEEE

3.3 Consistency control

Consistency control of synchronous events in our system is based on a token passing mechanism. An application replica is able to generate and send events to the peers only if it has a logical token provided by the platform. This is the default status for the cooperative 3D design application. If the replica is not the token owner, it can only receive events from the platform. However, this consistency control rule only applies to synchronous events. Isochronous events requiring real-time transmission, for example tele-virtual-cameras or audio data, can be generated and sent with no restriction.

The control mechanisms can either be transparent to the users and negotiated only between the application layer and the SM layer. Or, they may depend on the users' willingness regarding to the token passing. In our 3D cooperative design system an explicit token request, explicit token grant policy has been designed. Some other policies are under consideration for further development, such as providing several granularity levels within the same shared object or possibility of re-definition of different levels of optimism for a given interaction. For example, in a shared 3D workspace, some users may be interested in the creation of several locks for different 3D objects, allowing several users working and editing on the same shared environment. The platform is flexible enough for the inclusion of these policies, as they are simply a generalization of the existing mechanisms.

In summary, each replica of the 3D design system is, at any moment, in one of three possible states: passive (P), active (A) and intermediate (I). In the P state, it simply receives messages for local processing. In general, these messages contain user events. In the A state, the design program at one site can generate and send events to the peers. The I state represents the intermediate situation of a 3D design program that has requested the token but it has not yet been granted by the platform. During this period, the SM layer entities negotiate the exchange of states in the relative design context.

3.4 Service interfaces

When a cooperative working section is called, the cooperative platform is initialized by a cooperative working session initiator. The connections between all the cooperative working sites follow a client-server paradigm. The first phase is to establish a multipoint configuration. Either several point-to-point connections or multicast network protocols will be used among all the Group Communication (GC) entities. An algorithm between the initiator and all the other GC's will then be executed. Following this procedure, the Session Managers initialize themselves in a similar way [5].

A set of service primitives with the standard OSI structure are provided between layers for service request and supply. They are: srv.REQUEST, srv.RESPONSE, srv.INDICATION and srv.CONFIRM. These services may be confirmed, not confirmed or generated within a layer as a response to an internal condition.

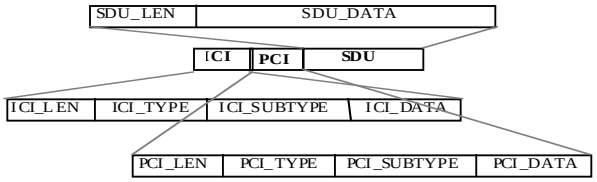

Figure 4. Structure of the Interface Data Unit for inter-process messages exchange

Figure 4 shows the structure of the data units that are transmitted between layers in each replica of the system and between cooperative working sites. They are the essential mechanism to request services and carry the exchanged data. These data units are called Interface Data Units (IDUs). They contain three components: Interface Control Information (ICI) component, Protocol Control Information (PCI) component and Service Data Unit (SDU) component.

The Interface Control Information component (ICI) represents the type of primitive being invoked by the application layer, including any additional parameters. This component has several parts: the type, subtype and optional data concerning the service being requested.

The primitives used for application data transfer (the data transfer for the 3D design system) include: SM_DATA.REQUEST/INDICATION, SM_PP_DATA.REQUEST / INDICATION, SM_RT_DATA.REQUEST / INDICATION.

The Group Communication entities (the GC's) have the corresponding services as GC service interface. They are: GC_DATA.REQUEST / INDICATION, GC_PP_DATA.REQUEST/INDICATION, GC_RT_DATA.REQUEST / INDICATION, etc.

The second component PCI contains the protocol information specific to the layer providing the service. The protocol specifying our particular application, the 3D design system will be defined in the Section 4 in the paper. The third component Service Data Unit SDU contains the data generated at the layer requesting the service. Further description of SDU will also be described in Section 4. The data unit is distributed to the peer entities at the same user layer located at each site . The combination of PCI + SDU is named a Protocol Data Unit (PDU). Some service

requests contain dummy PCI or SDU components because they refer to session control tasks only. In this case, the ICI component is issued with an optional set of parameters specifying the type of session control operation being requested.

The SM entities provide several SAPs (service access points) to the applications through sockets created during the connection requests. Connection establishment between the application layer and the SM layer follows the client-server paradigm with an additional set-up negotiation such as type of consistency control required, or any communication priorities.

4. The 3D Design Tool

CODI is defined as a multi-party 3D design system supported by the communication block with the distributed architecture.

4.1 The Scene

A CODI scene is a generic Open Inventor [6] object hierarchy. It is designed to be compatible with VRML 2.0. It is constructed by typical Inventor nodes and Deformable objects

The system is oriented to real-time WYSIWYG 3D scene edition and animation. The system provides support for 3D scene design where 3D object models of different formats can be inputted. To illuminate a scene the user can locate arbitrary number of different types of lights. The linear transformations such as rotation, scaling and translation and many other parameters of the objects, can be modified and viewed interactively. The parameters for modification include material information and textures. To help 3D scene designers to align a large number of objects, a collision detection option is provided [7]. It can keep the objects to touch with each other as close as possible without penetration. To allow complex modeling, a hierarchical system in the form of a tree has been implemented. The nodes of the tree are references or basic geometric objects. A reference can join different objects and other references to define more complex objects.

Animation capabilities are also provided by the system, allowing users to define animation parameters of moving objects. The aim is to give the user full freedom to animate any visible parameters of the virtual objects in the scene, not only for motion but also for object and lighting properties such as deformation, material, transparency, textures, color, etc.

The user interface is programmed using X11/Motif and Open Inventor Toolkit in C++, the system is also VRML compatible.

4.2 The user interface

A user-friendly interface for the user is essential for a complicated interactive system as CODI. Our user interface is designed to have three major parts: a tool bar on the top, an viewing area occupying most of the screen for manipulation and visualization of the 3D scene and a message area on the bottom for error messages or other help messages. In most of the cases there is only one window on the screen. When it becomes necessary, the user can open as many window as he likes [10].

The user interface has all the commands to support the 3D scene manipulation, visualization, groupware operation, audio-video control etc. The commands are designed in a pop-up manner. A list of elements appear in the tool bar on the upper part of the user interface. The tool bar has the following elements: scene, objects, lights, camera, windows, and groupware. When an element is pointed by the mouse, the manual showing the related operation buttons associated with this element will pop up. According to different operations, some floating window panels will appear for the user to choose the parameters of the operation. The user interface is shown in figure 5.

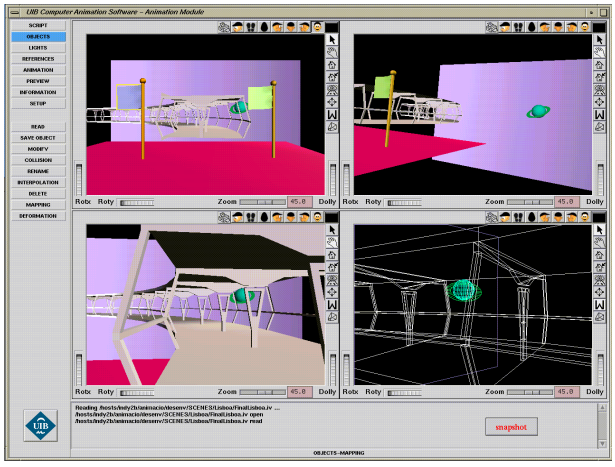

Figure 5. The user interface of the CODI system

4.3 The SMItoSMI protocol

As mentioned in Section 3, for multiple users to work cooperatively within the context of 3D scene design, an application specific protocol has to be defined on the distributed architecture. Each peer 3D designer has his own Session Manager Interface (SMI) in the CODI replica on his own workstation as indicated in Fig. 2. [8,9]

The SMItoSMI protocol governs the group interaction within the common 3D scene and is implemented through the exchange of SMItoSMI Protocol Data Units (PDUs)

between the peers. The Service Access Points (SAP) are used to access the groupware delivery service of the platform.

These PDUs encapsulate all the available cooperative 3D design functionality: multi-user application initialization, set-up of a common new 3D scenario, consistent multicasting of 3D object, light, virtual camera operations with 3D navigation capabilities.

The format of SMItoSMI PDUs comprises an integrated word that encodes the multicast event exchanged among the peer SMI's, and an application Service Data Unit (SDU) that encapsulates relevant parameters of the specific multicast event.

Here is a short table that lists our definition of the SMItoSMI PDU syntax.

Event	PDU Syntax	Service Requested to SM
USER INIT	UserInit(<userId>, <URL: user avatar>, <userContext>)	Reliable, Multipoint
USER NOTIFY	UserNotify(<userId>, <URL: user avatar>, <userContext>)	Reliable, Multipoint
NEW SCENE	NewScene(<userId>, <URL: iv scene>)	Reliable, Multipoint
CAMERA UPDATE	Camera_update(<userId>, <new camera parameters>)	Best-effort, Multipoint
ATTACH OBJECT	AttachObject(<userId>, <URL>)	Reliable, Multipoint
DETACH OBJECT	DetachObject(<userId>, <URL>, <element instance nr>)	Reliable, Multipoint
SELECT OBJECT	SelectObject(<userId>, <URL>, <element instance nr>)	Reliable, Multipoint
MODIFY SELECTED	ModifySelected(<userId>, <element type>, <modify code>, <new value>)	Reliable, Multipoint

The following events refer to token control and therefore are only relevant to the interface between the SMI and SM adjacent layers.

Event	IDU Syntax	Service Requested to SM
TOKEN REQUEST	TokenRequest()	Reliable, Point-to-point
TOKEN GRANT	TokenGrant()	Reliable, Point-to-point

The following sections explain the SMI protocol syntax by grouping the coded events into groups.

4.3.1 User Set-up and User Context

Each potential user that wishes to access to a multi-party 3D virtual environment (also named as a CODI working session), has to be previously registered in an authentication file. It is unique to all hosts of the CODI system. The register procedure creates a new entry in the authentication file:

```
<host ip name> <user name> <user id> <reserved parameter> <APPtoSM port nr> <SMtoGC port nr> <GCtoGC port nr>
```

where:

- The <host ip name> is the internet IP name of the host
- <user name> the name of the user
- <user id> is a unique identifier id number of a user
- <reserved parameter> specifies the type of host (0: Ordinary Host; 1: High Performance CPU Host)
- <APPtoSM port nr> is the SAP port number for full-duplex communication between its SMI element and the SM element
- <SMtoGC port nr> is the SAP port number for full-duplex communication between the local SM element and the local GC element
- <GCtoGC port nr> is the SAP port number the local GC element for a full-duplex communication with all the remote peer GC elements.

Any user registered in the CODIMembers authentication file may participate in a multi-party working design session. The initiative, the initiator to start such a session can be any registered user. It issues an initialization multicast event, USER_INIT to send to all other chosen peers. In response to the received USER_INIT event, each peer sends its own USER_NOTIFY event. After this handshaking that requires a reliable communication service, each replica of the CODI application is started in each participant's host.

Both type of initialization messages sent and received by peer users (USER_INIT and USER_NOTIFY), will include parameters specially devoted to the set-up of a multi-user space. They will enable each replica of CODI to build a dynamic list of remote users. This list creates the basic tool to support an abstraction of several users to interact synchronously or in real time.

Each entry of the list includes the remote user unique id number (taken from the CODIMembers file) and an application specific user context. It includes:

- A pointer to a SoSeparator node describing this remote user's active virtual camera.
- A personal “avatar” of the remote user. It is identified by his internet URL with a reference to an Open Inventor file containing the geometry description of the avatar. A pointer to the SoSeparator node of the local scene hierarchy is also contained pointing to the local copy of the avatar geometry.

- The indication of the CPU performance status of the remote user's host: a High Performance CPU Host (HPCH) or an Ordinary Host (OH),
- Other implementation specific information.

4.3.2 Scene Initialization and Token Control

When a working session begins, the initiator issues a multipoint NEW_SCENE event. The scene is identified by an internet URL. After the receipt of a NEW_SCENE message, each replica of CODI retrieves the scene from the specified URL using a dedicated point-to-point file transfer service. The object descriptions in the scene are then retrieved from the object URL's by the same sort of transfer service.

By the token control policy the system ensures that any modifications made by the token owner in the 3D scene, are coded and transmitted to all the peer participants. If other users desire to modify the 3D scene in any means, they need to request the token explicitly by issuing the TOKEN_REQUEST event. One of the users can do so only after the token is explicitly granted by receiving the TOKEN_GRANT event issued by the current token owner.

4.3.3 Scene Modification Messages

Possible operations to manipulate or modify a scene are to attach, detach, select and modify an object. Here the object is an Inventor object. It can be a geometrical object, a light source or a camera. Therefore, the possible modification to a 3D scene can be its illumination light sources, the parameters of the virtual camera to view the scene, the geometrical objects and their attributes etc.

Concerning scene illumination, the token owner can locate an arbitrary number of different types of lights by issuing the ATTACH_OBJECT event with the object type to be a light source. Default values of each type of light source will be stored and retrieved when an ATTACH_OBJECT event is issued. To change their intensity, color, lighting area etc., a MODIFY_SELECT_OBJECT event will be issued with an object type light source. A floating panel window will appear on the token holder's viewing area of his user interface. The modified light parameters will then be multicast to the peer session participants.

Primitive objects can be added to a scene at any time. Generic objects can also be retrieved from an Inventor file. In both cases a multicast ATTACH_OBJECT event will be issued by the token holder. When this type of events are generated, the SMItoSMI protocol works in a similar way as in the case of the NEW_SCENE event. Upon the receipt of an ATTACH_OBJECT event by the peers. the Inventor definition of the object itself, is later retrieved from the URL specified in the parameters of the ATTACH_OBJECT PDU.

While in the 3D scene, objects can be picked (creating SELECT_OBJECT events), deleted (generating DETACH_OBJECT events) and modified by linear transformations, such as rotation, scaling and translation, producing MODIFY events. In general, objects are recognized by a URL, locating an Inventor file with the object description, and an instance number.

Scene lighting properties for rendering of selected objects as well as the colors of lights, can be modified interactively, generating MODIFY_SELECTED multicast events. The parameters for object modification include material information, such as diffuse, specular and emissive colors, transparency, specular shininess and textures. For texture definition, the token owner can choose the texture parameters from a floating window panel. The texture coordinates can be plane, environment, cube, sphere or cylinder. He can also choose the way to combine the texture image and object color as modulation, blending, decal etc.

5. Conclusions

We have presented a collaborative 3D design system with the capability of synchronous multi-user, multi location 3D interaction in a common 3D virtual environment. The system is compliant with the Open Inventor and VRML 2.0 formats. It is a general system that can be applied to different scenarios where real time group interaction with 3D objects and spaces are necessary. Examples of such applications are collaborative product design, group development of urban planning and architect projects, or virtual studios for television program production.

This design system uses the services provided by a generic platform that is implemented on top of existing network and transport protocols. This layered approach separates the application and communication functions, hiding the underlying configurations from the user application. Different network technologies and protocols may be used without changing the service primitives at the application level. At the moment, real-time services are provided on top of best-effort transport and/or network protocols, due to the characteristics of the internetworking environment tested so far. This means that some QoS specifications may not be guaranteed under heavy network conditions. The system has been designed and partially implemented. It will take more time to complete all the functions of the system.

Acknowledgment

The work is supported by the CICYT funding TEL 96-0544 and IN 96-0151. It is also a close cooperation between the UIB in Spain and ADETTI in Portugal.

References

[1] T. Crowley et al., "MMConf: An infrastructure for building shared multimedia applications," Proc. CSCW 90, ACM Press, New York, 1990, pp. 329-342.

[2] Yavatkar, R. and Lakshman, K., "Communication support for distributed collaborative applications," Multimedia Systems, 2: 74-88, Springer-Verlag.

[3] Mills, D. L., "Improved Algorithms for Synchronizing Computer Network Clocks," ACM SIGCOMM´94 Conf. Proceedings, Oct/94.

[4] Greenberg, S. and Marwood, D. (1994): "Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface," ACM CSCW´94 Conf. Proceedings, Chapel Hill, North Carolina, Oct/94, USA.

[5] Almeida, A. and Belo, C. A., "Support for Multimedia Co-Operative Sessions over Distributed Environments", Proc. MEDIACOMM'95, SCS (Society for Computer Simulation), Southampton, April/1995

[6] Werneck, J., "The Inventor Mentor, Programming Object-Oriented 3D Graphics with Open Inventor Release 2," Open Inventor Architecture Group, Addison-Wesley 1994.

[7] Galli, R., *et. al*., "Real-time Collision Checking for 3D Objects Positioning in Sparse Environments," Proceedings of the European Workshop on Combined Real and Synthetic Image Processing for Broadcast and Video Production, Hamburg, 23-24 November 1994.

[8] J. M. Dias, R. Galli, A. C. Almeida, C. A. C. Belo, J. M. Rebordao. "mWorld: A Multi-user 3D Virtual Environment with Synchronous Communication Support." IEEE Computer Graphics and Applications, March-April 1997.

[9] Dias, J. M., Fallon, N., Almeida, A. C., McGuinness, F., Hofmann, J. and Belo, C. A. (1994), "FASHION-NET, A Collaborative Multimedia Design System for the Apparel Industry," in W. Bauerfeld, O. Spaniol and F. Williams (eds.): Broadband Islands '94: Connecting with the End-User, Elsevier Science B. V.

[10] M. Mascaró, P. Palmer, Y. Luo, "Generación de entornos virtuales para fusión de imagen real y sintética," CEIG´96, junio, 1996, Valencia, España.